<

DEFINIÇÃO

Apresentação da evolução histórica dos bancos de dados, abordando as características dos sistemas de banco de dados (SBD) e a arquitetura dos sistemas de gerência de banco de dados (SGBD).

PROPÓSITO

Compreender a origem e as características dos SBDs, bem como suas funcionalidades, vantagens e desvantagens, além de conhecer a arquitetura dos SGBDs e os sistemas mais utilizados em bancos de dados.

OBJETIVOS

MÓDULO 1

Reconhecer o histórico dos bancos de dados e suas tecnologias

MÓDULO 2

Identificar as características dos sistemas de banco de dados (SBD)

MÓDULO 3

Descrever a arquitetura dos sistemas de gerência de banco de dados (SGBD)

MÓDULO 1

- Reconhecer o histórico dos bancos de dados e suas tecnologias

DEFINIÇÃO DE BANCO DE DADOS

Você certamente já leu sobre o termo **banco de dados** em algum contexto técnico ou geral na mídia tradicional ou na internet.

Mas o que é um banco de dados?

O termo banco de dados, no sentido técnico, origina-se de ***database***, do inglês, e o livro-texto de edição norte-americana mais adotado no mundo o define de maneira simples e direta: “banco de dados é uma coleção de dados relacionados” (ELMASRI; NAVATHE, 2019), em que dados são **fatos conhecidos que podem ser registrados e possuem significado implícito**.

★ EXEMPLO

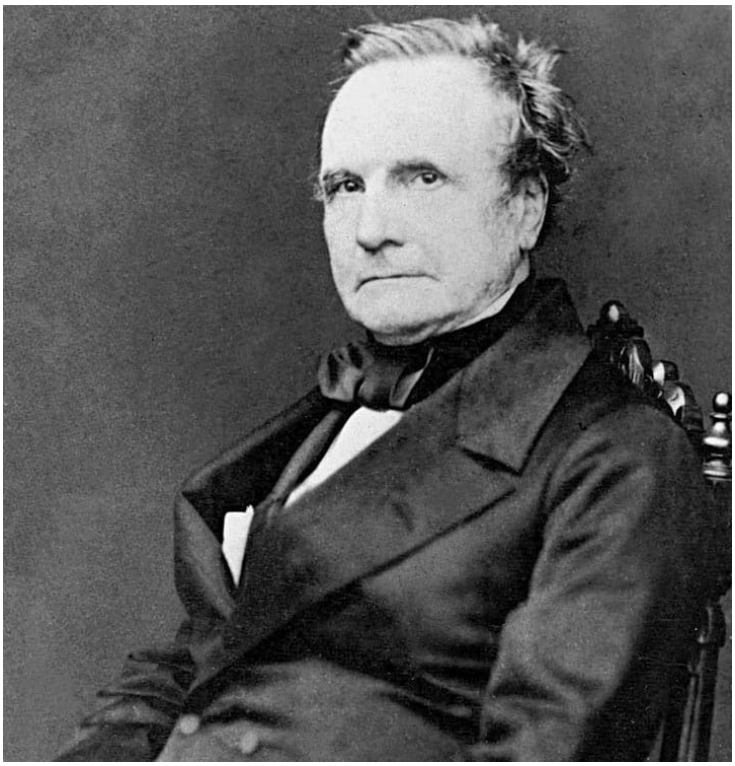
Nome, data de nascimento, endereço e telefone são dados relacionados entre si, inerentes a uma pessoa conhecida em certo contexto. Antes de elaboramos mais essa definição, vamos relembrar um pouco da história dos bancos de dados.

Fonte: Shutterstock | **Por**: fullvector

O conceito de banco de dados como uma coleção de dados relacionados sempre existiu como componente central dos sistemas de informação. Estes, por sua vez, sempre existiram desde que a humanidade se organizou como sociedade. Segundo afirmam Melo, Silva e Tanaka (1998), o que tem mudado rapidamente ao longo da história é a tecnologia que permite a sua implementação e que se confunde com o próprio conceito de sistemas de informação.

Assim, antes da existência do computador, bancos de dados existiam sob a forma de registros físicos em papel, organizados em pastas dentro de armários, que formavam os arquivos dos sistemas de informação, operados manualmente pelos seus usuários. Será que ainda existem sistemas de informação desse tipo em pleno século XXI?

EVOLUÇÃO DOS SISTEMAS DE INFORMAÇÃO EM COMPUTADOR



Fonte:Shutterstock

Charles Babbage. (**Fonte**: Wikimedia)

ERA DO PROCESSAMENTO DE DADOS

Historicamente, o computador, inventado na década de 1940 ao fim da Segunda Guerra Mundial, era usado primordialmente como uma máquina para cálculos matemáticos complexos, a exemplo da máquina diferencial de **Charles Babbage**, do século XIX.

CHARLES BABBAGE

Charles Babbage (1791-1871) é um dos mais célebres ícones no mundo da computação. As suas notáveis contribuições para a área fizeram dele o pioneiro dos computadores. A Máquina Diferencial N. 1 foi a primeira calculadora automática a ser inventada e ainda é considerada uma peça única pela precisão que na época apresentava.

Fonte: Pplware.

Logo se percebeu que, graças à arquitetura criada pelo seu inventor, **John von Neumann**, baseada em uma unidade central de processamento que armazena programas e dados, o computador também serve para o processamento de dados e não apenas para cálculos.

Fonte:Shutterstock

John von Neumann. (**Fonte**: Los Alamos National Laboratory)

JOHN VON NEUMANN

John von Neumann (1903-1957) foi um matemático húngaro, naturalizado norte-americano, considerado o pai da Teoria dos Jogos. Seus interesses abrangiam lógica, assuntos militares, computação, economia, entre outros.

Fonte:Shutterstock

Fonte: Shutterstock | **Por**: Lifestyle and Wedding ph

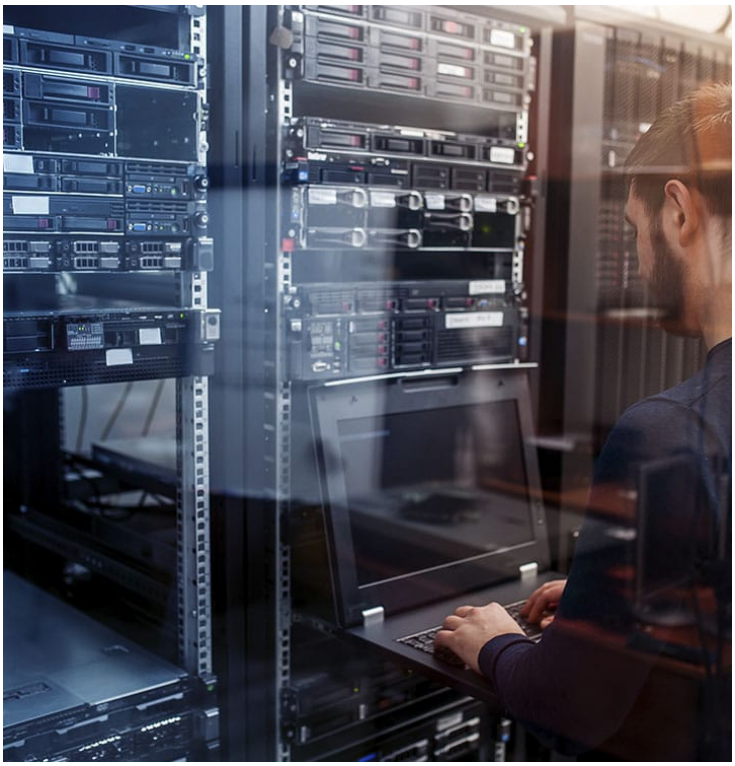
Essa utilidade do computador foi impulsionada com a invenção do disco magnético pela IBM, em 1957, que o denominou de Dispositivo de Armazenamento de Acesso Direto (DASD, do inglês ***Direct Acess Storage Device***).

Fundamentalmente, o que esse dispositivo – atualmente conhecido como disco rígido e pela sigla HD (***Hard Disk***) – apresentou de novidade, à época, foi a capacidade de leitura de dados externos à unidade central de processamento de forma direta, sem a necessidade de leitura sequencial, como em fitas magnéticas.

Com o advento do armazenamento externo em disco rígido, nasceu a era do processamento de dados por computador.

Você já ouviu falar em Centro de Processamento de Dados (CPD), denominação que ainda persiste em organizações tradicionais?

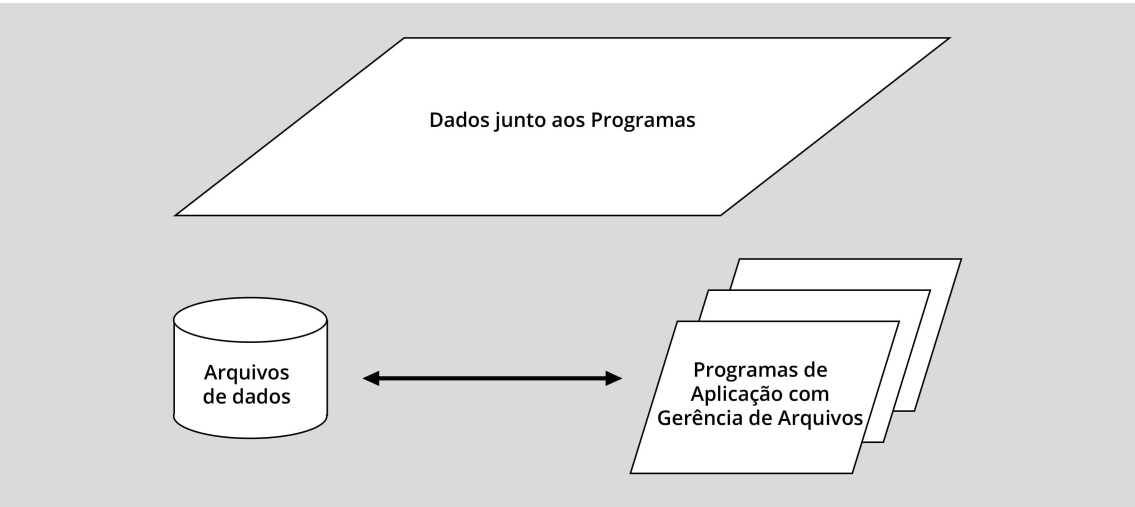
Nessa era, os programas de aplicação, desenvolvidos em uma linguagem de programação (usualmente COBOL, em aplicações empresariais, ou Fortran, em aplicações científicas), manipulavam dados armazenados em arquivos hospedados em disco magnético, utilizados pelo sistema operacional e formando o que se denomina **sistema de arquivos**.



Fonte:Shutterstock

Fonte: Shutterstock | **Por**: Alexandru Chiriac

A figura a seguir mostra a evolução obtida ao possibilitar que os programas acessassem os dados externamente em arquivos no disco (configurando um avanço em relação aos programas que continham internamente os próprios dados) para execução em lotes (***batch***, em inglês), na fase inicial do uso do computador.



Fonte: Tanaka (2018)

Esse modelo de processamento de dados com sistema de arquivos foi largamente utilizado no início do emprego do computador em sistemas de informação empresariais, após o advento do disco magnético, persistindo até os dias atuais, nos chamados sistemas legados. Exemplo disso, foi a maior demanda por programadores da linguagem COBOL durante a pandemia de

COVID-19 para realizar manutenção em sistemas da Administração Pública do governo dos EUA.

SAIBA MAIS

Para saber mais sobre o aumento da procura por programadores COBOL durante a pandemia de COVID-19, não deixe de verificar a indicação feita no Explore + ao fim deste tema.

PRIMÓRDIOS DOS SISTEMAS DE BANCO DE DADOS

Fonte:Shutterstock

Seguindo na história, o advento dos bancos de dados foi uma evolução natural dos sistemas de arquivos. Observe na figura anterior que os programas os quais manipulam os arquivos de dados, além de implementarem a lógica da aplicação, têm de conter um módulo para a gerência dos arquivos de dados. Esse módulo deve ser repetido em todos os programas que precisam acessar e manipular o mesmo arquivo de dados.

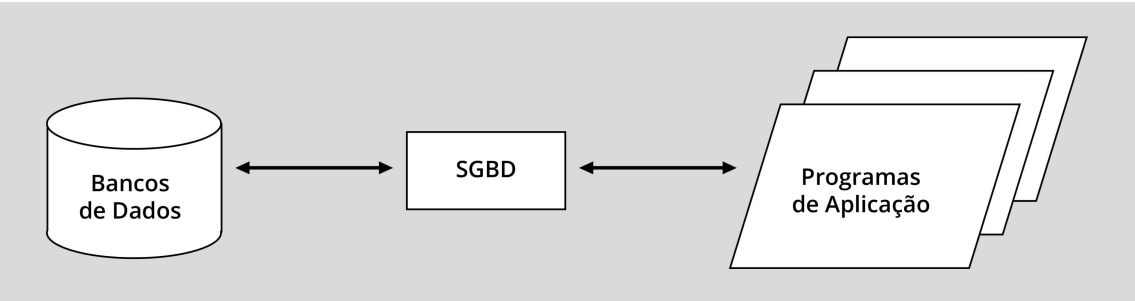
Fonte:Shutterstock

Por exemplo, o departamento de pessoal de uma organização mantém o arquivo com os dados dos empregados. Suponha que o departamento de produção também precise usar dados desse arquivo para alocar empregados em projetos. Nesse caso, os programas de aplicação que atendem aos dois departamentos deverão conter o mesmo módulo de gerência do arquivo de empregados, causando uma repetição de código de programação e dificultando a sua manutenção.

Fonte:Shutterstock

Assim, os **sistemas de banco de dados** (SBD) vieram para mitigar esse problema, a partir de 1960, tirando dos programas de aplicação a responsabilidade de gerenciar os arquivos de dados, tarefa que passou a ser delegada a um software intermediário, denominado de sistema de gerência de banco de dados (SGBD), como mostra a figura a seguir.

Essa propriedade dos sistemas de banco de dados é denominada de **independência entre dados e programas**, uma diferença primordial em relação aos sistemas de arquivos.



Fonte: Tanaka (2018)

Em outras palavras, ocorreu uma modularização do sistema de informação, com a distribuição de responsabilidades entre os programas de aplicação e o SGBD. Os programas de aplicação passaram a se ocupar exclusivamente das funcionalidades da aplicação propriamente dita, deixando as tarefas de acesso e manipulação dos dados armazenados em disco para o SGBD, um software tipicamente auxiliar, de bastidores ou, como se costuma dizer no jargão do mercado, um serviço de ***back end***.

📢 ATENÇÃO

Perceba a diferença entre o sistema de banco de dados (SBD) e o sistema de gerência de banco de dados (SGBD), pois o primeiro é mais amplo, englobando o SGBD, os próprios programas de aplicação e os bancos de dados manipulados por eles.

Neste ponto, cabe um questionamento importante, cada vez mais válido em face dos avanços das tecnologias de hardware de memória, tanto de memória principal (RAM) quanto de memória secundária (discos magnéticos ou ***hard disk drives*** (HDD) e de semicondutores ou ***solid state drives*** (SSD)).

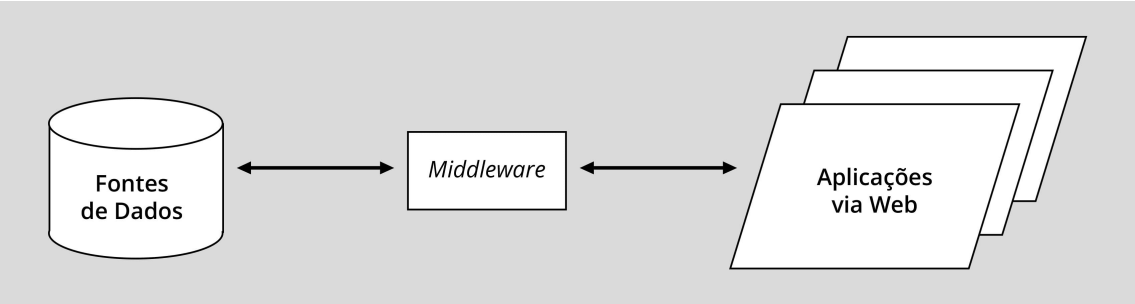
A questão é: qual dos três modelos de sistemas é o mais eficiente para uma aplicação com o mesmo volume de dados, ou seja, o modelo monolítico com dados junto dos programas; o modelo com sistemas de arquivos; ou o modelo com sistemas de bancos de dados?

Adiaremos a resposta a essa questão para o final deste módulo.

ESTÁGIO ATUAL DOS SISTEMAS DE INFORMAÇÃO (NA WEB)

Antes de discorrer sobre o histórico dos SBDs, vale completar a evolução dos sistemas de informação até os dias atuais, fortemente influenciada pela revolução tecnológica causada pela ***World Wide Web*** no final do século XX.

Com a popularização da interface Web no desenvolvimento das aplicações, surgiram novas linguagens de programação e novas formas de armazenamento e acesso a dados em fontes com diferentes formatos. Assim, o SGBD das aplicações tradicionais pode ser considerado atualmente como um gênero de software básico, com papel intermediário, que denominamos na figura a seguir de ***middleware***, em que se incluem servidores de aplicações das diferentes linguagens e ambientes de desenvolvimento Web.



Fonte: Tanaka (2018)

Essa figura resume o atual estágio dos sistemas de informação na Web, em que as fontes de dados não se restringem a dados estruturados, como em bancos de dados tradicionais, admitindo volumes gigantescos em diversos formatos, localizações e velocidade de produção, características marcantes do conceito de ***Big Data***. Igualmente, as aplicações via Web são desenvolvidas em uma diversidade de plataformas digitais, de ***smartphones*** a supercomputadores, que têm em comum a conexão com a internet e, em consequência, a computação em nuvem (***Cloud Computing***).

EVOLUÇÃO DOS SISTEMAS DE BANCO DE DADOS

BANCOS DE DADOS NAVEGACIONAIS

Há uma controvérsia sobre qual foi o primeiro SGBD implementado e utilizado comercialmente na década de 1960. Sabe-se que duas iniciativas independentes ocorreram paralelamente, resultando em dois produtos comerciais:

IDS (*INTEGRATED DATA SYSTEM*)

Criado por Charles Bachman (1924-2017) no âmbito de um comitê que padronizou a linguagem COBOL (CODASYL, de ***Committee on Data Systems Languages***).

IMS (*INFORMATION MANAGEMENT SYSTEMS*)

Criado pela IBM na esteira do sucesso da invenção do disco magnético anos antes.

O IDS e o IMS tinham em comum a característica de que os dados eram acessados por meio de programas que “navegam” de registro em registro pela estrutura dos dados armazenados em disco. Por causa dessa característica, atualmente aqueles SGBDs e outros que seguiram a mesma abordagem são denominados de navegacionais.

Observe a **diferença** entre eles:

IDS

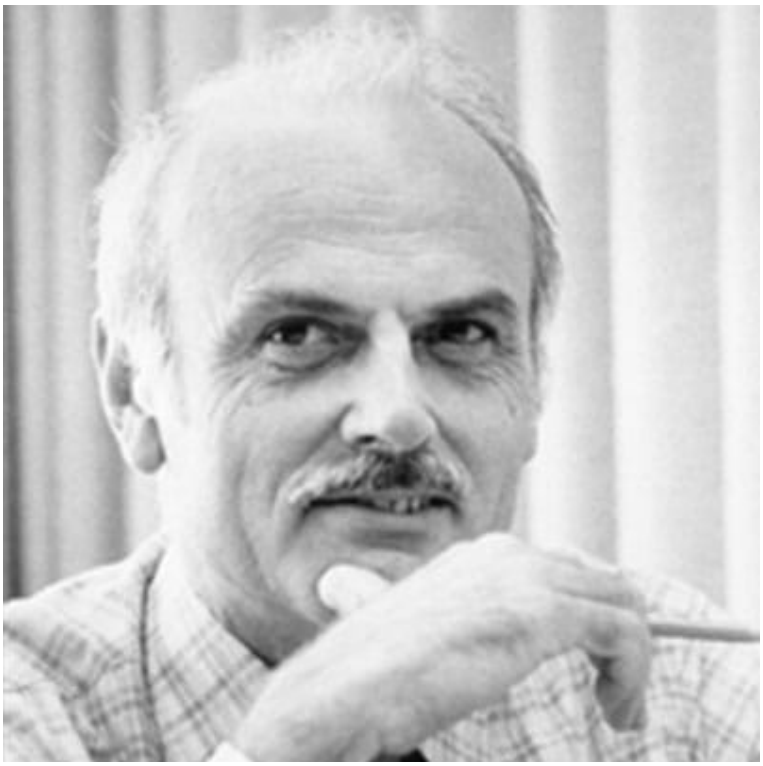
Usava a estrutura de dados de grafos ou redes, daí a denominação de ***network databases***.

IMS

Adotava a estrutura de dados de árvores, que é um tipo de grafo mais restrito do que as redes, baseado em hierarquias, originando a denominação ***hierarchical databases***.

Vários SGBDs foram implementados com variantes desses modelos de banco de dados, como o DMS (***Data Management System***) e o IDMS (***Integrated Database Management System***). Vale relembrar que muitos sistemas de informação legados daquela época ainda utilizam esses SGBDs navegacionais, a exemplo da demanda por programadores COBOL em meio à pandemia de COVID-19.

O MODELO RELACIONAL DE BANCO DE DADOS



Fonte:Shutterstock

Edgar F. Codd. (**Fonte**: Wikimedia)

A grande revolução na história dos bancos de dados ocorreu na virada das décadas de 1960 e 1970, com a publicação do artigo seminal do matemático pesquisador da IBM, **Edgar Codd**, intitulado *A Relational Model of Data for Large Shared Data Banks*, que introduziu o modelo relacional de banco de dados.

O artigo de Codd, uma das obras mais citadas na comunidade da computação em todos os tempos, foi o marco do chamado modelo relacional de banco de dados, cuja estrutura de dados, diferentemente dos grafos dos bancos de dados navegacionais, é uma função matemática denominada relação.

EDGAR CODD

Edgar Frank Codd (1923-2003) foi um cientista da computação e matemático americano que inventou o modelo de dados relacionais, que levou à criação do banco de dados relacional, um método padrão de recuperação e armazenamento de dados do computador.

Fonte: Encyclopædia Britannica.

Codd criou uma Álgebra Relacional e um Cálculo Relacional, nos quais baseou toda a teoria matemática das relações em que fundamentou o modelo relacional. Apesar da base teórica do modelo, a estrutura de dados subjacente tem o mérito de ser muito simples, pois uma relação nada mais é do que uma tabela formada por colunas e linhas, em cujas células estão armazenados os dados, conceito compreensível pelo senso comum de qualquer leigo em Matemática ou computação, como podemos ver a seguir.

Linhas/Colunas	Nome	Data de nascimento	Sexo	Departamento
Linha 1	Jorge	15/01/1999	Masculino	Produção
Linha 2	Ana	05/07/1980	Feminino	Recursos Humanos
Linha 3	Joana	10/09/1990	Feminino	Pesquisa

Atenção! Para visualização completa da tabela utilize a rolagem horizontal

A solidez da fundamentação matemática do modelo relacional disparou uma série de iniciativas de implementação em empresas, como a própria IBM, e no meio acadêmico, principalmente nas universidades do estado da Califórnia, onde se localizava o centro de pesquisas da IBM. A partir de então, a IBM patrocinou o projeto **System R** (de ***Relational***), enquanto a Universidade da Califórnia em Berkeley (UCB) deu início à implementação acadêmica de um SGBD relacional denominado de **Ingres** (***Interactive Graphics Retrieval System***).

Fonte:Shutterstock

DB2. Fonte: IBM

PRINCIPAIS SGBDS RELACIONAIS

O projeto **System R** deu origem ao SGBD comercial da IBM, inicialmente denominado SQL/DS (***Structured Query Language/Data System***), depois renomeado de DB2, atualmente um dos líderes no mercado de bancos de dados corporativos, com versões em diferentes plataformas de hardware/software e na nuvem.

SAIBA MAIS

A linguagem SQL, criada pela IBM como uma linguagem de consulta e manipulação de dados dos bancos de dados relacionais, passou a ser conhecida como sinônimo de SGBD relacional, chegando a ser confundida com produtos que levam a sigla em seu nome.

No âmbito comercial, também despontou, como decorrência do sucesso do modelo relacional, o desenvolvimento de um SGBD pela empresa inicialmente denominada Software Development Laboratories (SDL), depois renomeada Relational Software Inc. (RSI) e, finalmente, Oracle Corporation, nome pelo qual é atualmente reconhecido como o SGBD líder do mercado global de banco de dados.

Fonte:Shutterstock

Fonte: Shutterstock | **Por**: Anton Garin

Fonte:Shutterstock

Fonte: Shutterstock | **Por**: dennizn

Em 2010, com a aquisição da Sun Microsystems, uma grande empresa de hardware tradicionalmente incentivadora de projetos de software livre, a Oracle incorporou entre seus produtos o MySQL, um SGBD relacional de reconhecida liderança na comunidade de

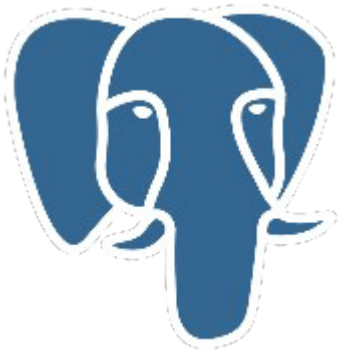
desenvolvimento de sistemas para a Web. O SGBD MySQL, associado ao sistema operacional Linux, ao servidor Web Apache e à linguagem de programação PHP, formou o quarteto de software conhecido pela sigla LAMP, de grande sucesso no desenvolvimento de aplicações Web até os dias atuais.

Na frente acadêmica, o projeto Ingres, da UCB (Universidade da Califórnia, Berkeley) deu origem a versões comunitárias mediante licença livre da própria universidade junto com seu sistema operacional Unix, denominado BSD (***Berkeley Software Distribution***). O projeto acadêmico originou um produto comercial de mesmo nome, Ingres DBMS, que concorreu diretamente com o Oracle e o SQL/DS nos primórdios dos SGBDs relacionais.

O esforço de desenvolvimento do Ingres envolveu muitos pesquisadores, professores e estudantes, os quais acabaram levando o seu código livre em linguagem C para implementação em outros produtos comerciais, notadamente o SGBD Sybase que, na década de 1990, associou-se à Microsoft, dando origem ao SQL Server, atualmente um dos líderes no mercado de bancos de dados relacionais.

Fonte:Shutterstock

Fonte: Wikipedia.



Fonte:Shutterstock

Logo Postgre. Fonte: Wikimedia

A continuidade do projeto Ingres deu frutos também na área acadêmica com a evolução para um modelo de dados além do relacional, estendido com conceitos da programação orientada a objetos, denominado Postgres (de *Post Ingres*).

SAIBA MAIS

Após a incorporação da linguagem SQL na década de 1990, o Postgres foi rebatizado como PostgreSQL, atualmente reconhecido como o mais avançado SGBD ***open source*** do mundo. Para saber mais sobre o PostgreSQL, não deixe de verificar a indicação feita no Explore+ ao fim deste tema.

O PostgreSQL e o MySQL são os SGBDs mais utilizados no aprendizado dos bancos de dados relacionais pela sua popularidade e pelo fato de disponibilizarem versões com licença e documentação livres.

OUTROS MODELOS DE SGBDS

No *ranking* de popularidade dos SGBDs, disponibilizado pelo DB-Engines em seu website, destacam-se, entre os que adotam o modelo relacional: Oracle, MySQL, Microsoft SQL Server, PostgreSQL e IBM DB2.

Cabe observar que esse *ranking* não trata exclusivamente de SGBDs do modelo relacional de dados. Os próprios SGBDs relacionais, mencionados como líderes de mercado, são

classificados no *ranking* como “multimodelos”, porque implementam funcionalidades que vão além do modelo relacional. Vejamos:

ORACLE

Relacional e multimodelo (documentos, grafos e RDF)

MYSQL

Relacional e multimodelo (documentos)

MICROSOFT SQL SERVER

Relacional e multimodelo (documentos e grafos)

POSTGRESQL

Relacional e multimodelo (documentos)

IBM DB2

Relacional e multimodelo (documentos e RDF)

E O QUE SÃO ESSES OUTROS MODELOS DE BANCO DE DADOS, ALÉM DO RELACIONAL?

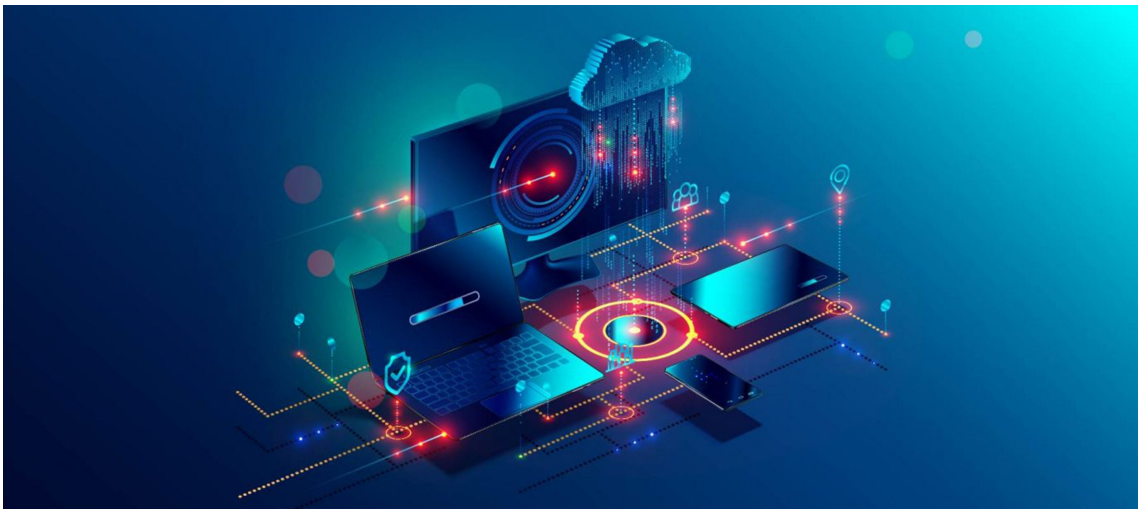
Não resta dúvida de que o modelo relacional se firmou no mundo corporativo, sendo utilizado na grande maioria dos sistemas de informação empresariais pela sua popularidade e robustez dos produtos disponíveis ao longo de décadas de desenvolvimento, bem como pela padronização e pelo uso da linguagem de consulta e manipulação de dados SQL.

Entretanto, existem aplicações em sistemas de informação que requerem muito mais recursos de armazenamento e manipulação de dados do que as tabelas do modelo relacional, em especial aplicações Web e de cunho científico que processam grandes quantidades de dados em formatos diversos, com as atuais tendências como ***Big Data***, ***Internet of Things*** e ***Data Science***.

Assim, vários modelos de banco de dados não relacionais vêm surgindo no mercado, sendo denominados de NoSQL, termo traduzido como “Não SQL” ou “Não somente SQL” (de ***Not Only*** SQL).

SÃO, DE FATO, BANCOS DE DADOS QUE NÃO ADOTAM O MODELO RELACIONAL DE DADOS E, PORTANTO, NÃO USAM A LINGUAGEM SQL, EMBORA ALGUNS POSSUAM IMPLEMENTAÇÕES DO COMANDO SELECT DA SQL PARA FINS DE COMPATIBILIDADE DE LINGUAGEM DE CONSULTA COM OS BANCOS DE DADOS RELACIONAIS.

O estudo de bancos de dados NoSQL está fora do escopo deste tema, constituindo-se em um tema à parte pela diversidade dos seus conceitos e de suas tecnologias.



Fonte: Shutterstock | **Por**: Andrey Suslov

É importante conhecer a importância dessa tendência dos SGBDs, como demonstra o site DB-Engines Ranking, que apresenta mais de uma dúzia de modelos de bancos de dados NoSQL com seus principais produtos, muitos deles multimodelos.

SAIBA MAIS

Confira a lista de multimodelos

Chave-Valor: Redis, Amazon DynamoDB, Microsoft Azure CosmosDB.

Documentos: MongoDB, Amazon DynamoDB, Microsoft Azure CosmosDB.

Séries temporais: InfluxDB, KDB+, Prometheus.

Grafos: Neo4J, Microsoft Azure CosmosDB, ArangoDB.

Orientado a objetos: InterSystems Caché. Versant Object Database, ObjectStore.

Motores de busca: Elasticsearch, Splunk, Solr.

RDF (Resource Description Framework): Marklogic, Apache Jena, Virtuoso.

Colunar: Cassandra, HBase, Microsoft Azure CosmosDB.

Multivalores: Adabas, UniData/UniVerse, jBASE.

XML nativo: Marklogic, Oracle Berkeley DB, Virtuoso.

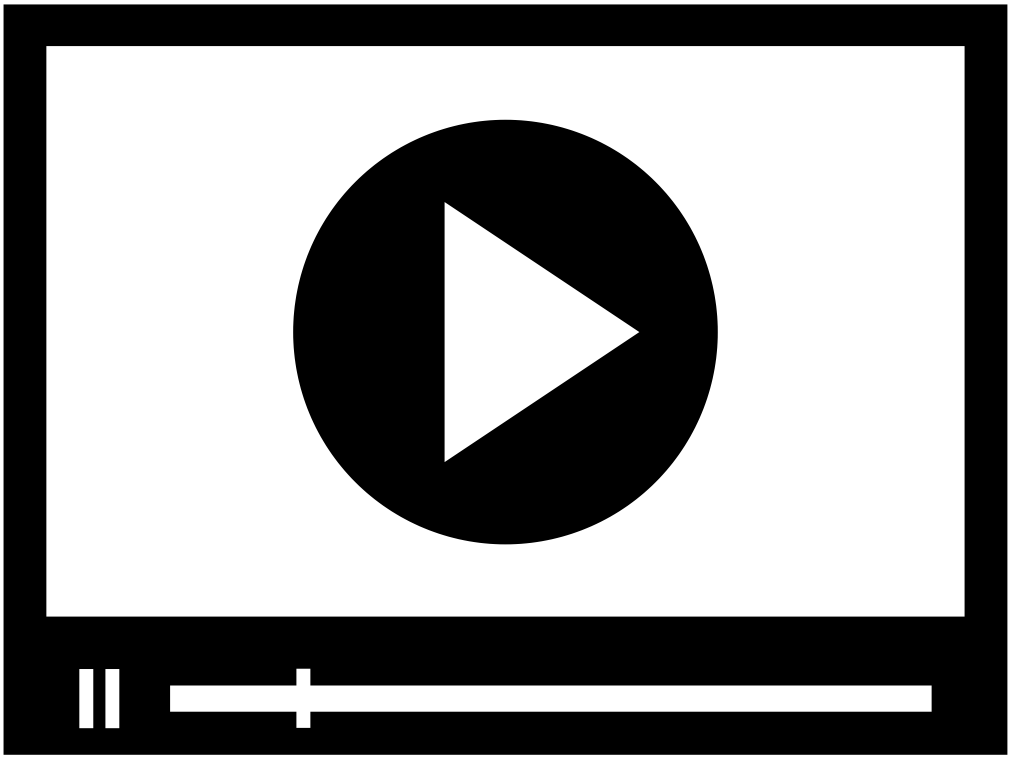
Eventos: Event Store, IBM DB2 Event Store, NEventStore.

Conteúdos: JackRabbit, ModeShape.

Navegacional: IMS, IDMS.

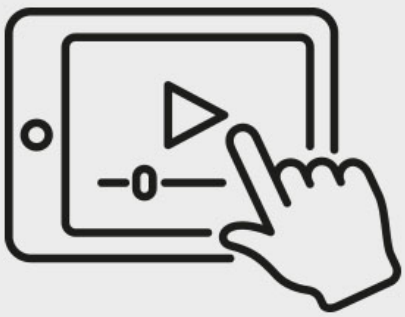
COMENTÁRIO

O modelo navegacional é exatamente aquele dos primórdios dos sistemas de banco de dados, da década de 1960, antes do advento do modelo relacional, cujos produtos ainda continuam sendo utilizados, principalmente em sistemas de informação legados daquela época.



Para fechar o módulo, neste vídeo o professor Sidney Ventury aprofundará o conteúdo falando um pouco mais sobre as vantagens que o Banco de Dados trouxe em relação ao antigo sistema de arquivos.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



VERIFICANDO O APRENDIZADO

1. AO FINAL DESTE MÓDULO, É UM BOM MOMENTO PARA RETORNAR À QUESTÃO LEVANTADA DURANTE A DESCRIÇÃO DOS MODELOS DE COMPUTAÇÃO EM SISTEMAS DE INFORMAÇÃO. QUAL MODELO É O MAIS EFICIENTE PARA UMA APLICAÇÃO NA MESMA LINGUAGEM DE

PROGRAMAÇÃO, COM O MESMO VOLUME DE DADOS, SOB A MESMA INFRAESTRUTURA DE HARDWARE?

A) Modelo monolítico com os dados junto dos programas.

B) Modelo de sistema de arquivos, com os programas acessando os dados através de módulos internos de gerência de arquivos.

C) Modelo de sistema de banco de dados, com os programas acessando os dados através de um SGBD.

D) Modelo de sistema de informação na Web, com os programas acessando os dados na nuvem.

2. QUAL O NOME DA LINGUAGEM DE CONSULTA E MANIPULAÇÃO DE BANCO DE DADOS QUE SE TORNOU SINÔNIMO DO MODELO RELACIONAL DE BANCO DE DADOS?

A) PostgreSQL

B) MySQL

C) SQL

D) SQL Server

GABARITO

1. Ao final deste módulo, é um bom momento para retornar à questão levantada durante a descrição dos modelos de computação em sistemas de informação. Qual modelo é o mais eficiente para uma aplicação na mesma linguagem de programação, com o mesmo volume de dados, sob a mesma infraestrutura de hardware?

A alternativa **"A "** está correta.

Consideradas as mesmas condições de aplicação, volume de dados e infraestrutura, conforme o enunciado, o modelo monolítico será mais eficiente, pois todo o processamento será

realizado na velocidade de processamento junto à memória principal (RAM), que é muito mais rápida do que o acesso à memória secundária ou à nuvem.

Na verdade, as alternativas dessa questão estão na sequência de eficiência dos modelos de computação sob as mesmas condições. Isso não significa que os bancos de dados são ineficientes, pois eles se aplicam adequadamente a cenários com volumes de dados que não cabem na memória principal, como é o caso de sistemas de informação corporativos.

2. Qual o nome da linguagem de consulta e manipulação de banco de dados que se tornou sinônimo do modelo relacional de banco de dados?

A alternativa **"C "** está correta.

Todas as demais alternativas referem-se a nomes de SGBDs que implementam o modelo relacional e, portanto, implementam a linguagem SQL (*Structured Query Language*), chegando a confundir o nome do produto com a linguagem.

MÓDULO 2

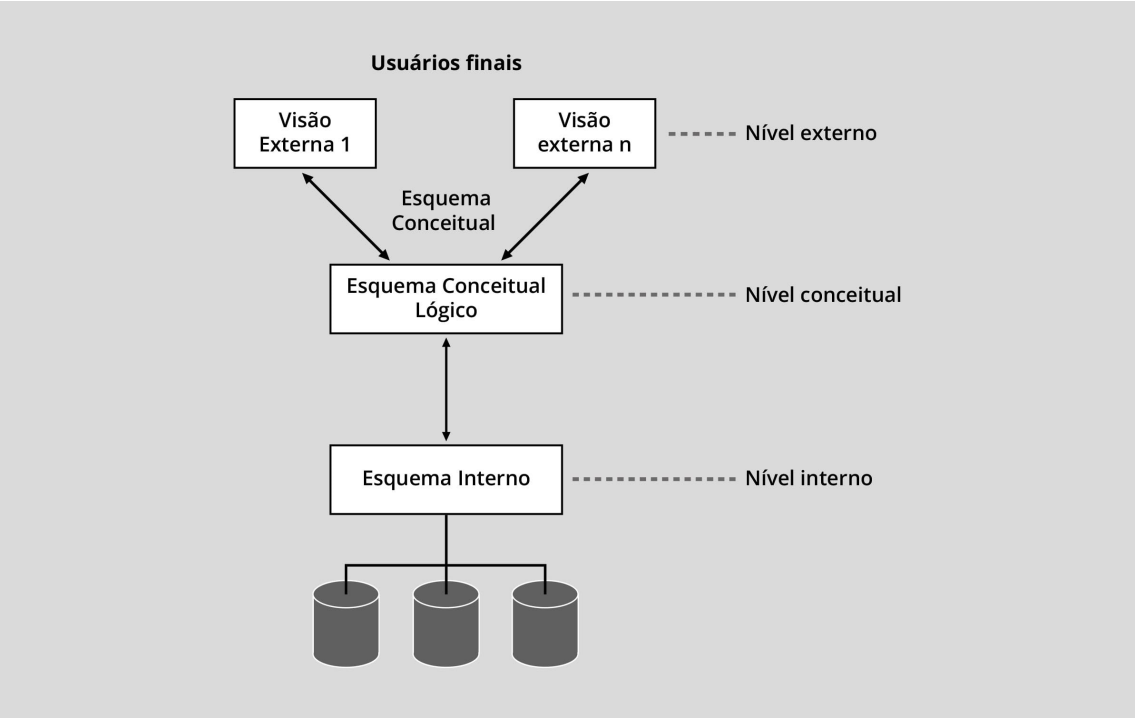
- Identificar as características dos sistemas de banco de dados (SBD)

DIFERENÇAS ENTRE SISTEMA DE ARQUIVOS E SISTEMA DE BANCO DE DADOS

No módulo anterior, verificamos que uma característica primordial que distingue os sistemas de banco de dados dos sistemas baseados de arquivos é a independência dos dados em relação a programas. Essa, na verdade, é apenas uma espécie da característica dos sistemas de banco de dados denominada independência de dados.

O conceito de independência de dados deriva da arquitetura de três esquemas do banco de dados que separa as aplicações dos usuários finais do banco de dados físico armazenado.

A figura a seguir é uma adaptação da arquitetura ANSI/SPARC, que considera três níveis de esquemas: o **nível externo**, que contém o esquema conceitual externo, integrando as diferentes visões dos usuários das aplicações; o **nível conceitual**, o qual contém o esquema conceitual lógico ou de implementação, descrevendo a estrutura lógica do banco de dados; e o **nível interno**, que contém o esquema interno ou físico do banco de dados armazenado em disco.



Fonte: Tanaka (2018)

Essa arquitetura serve para definir dois tipos de independência de dados, além da independência entre dados e programas já estudada:

Independência lógica de dados

Consiste na capacidade de se alterar o esquema conceitual lógico, por exemplo, acrescentando um item de dado, sem alterar o esquema conceitual externo, isto é, as visões externas dos usuários através dos programas de aplicação.

Independência física de dados

Consiste na capacidade de se alterar o esquema interno, por exemplo, reorganizando os arquivos físicos que armazenam os dados, sem alterar o esquema conceitual lógico e, em consequência, o esquema conceitual externo.

Outras características, entre muitas, que diferenciam os sistemas de bancos de dados dos sistemas de arquivos são descritas a seguir.

NATUREZA AUTOCONTIDA

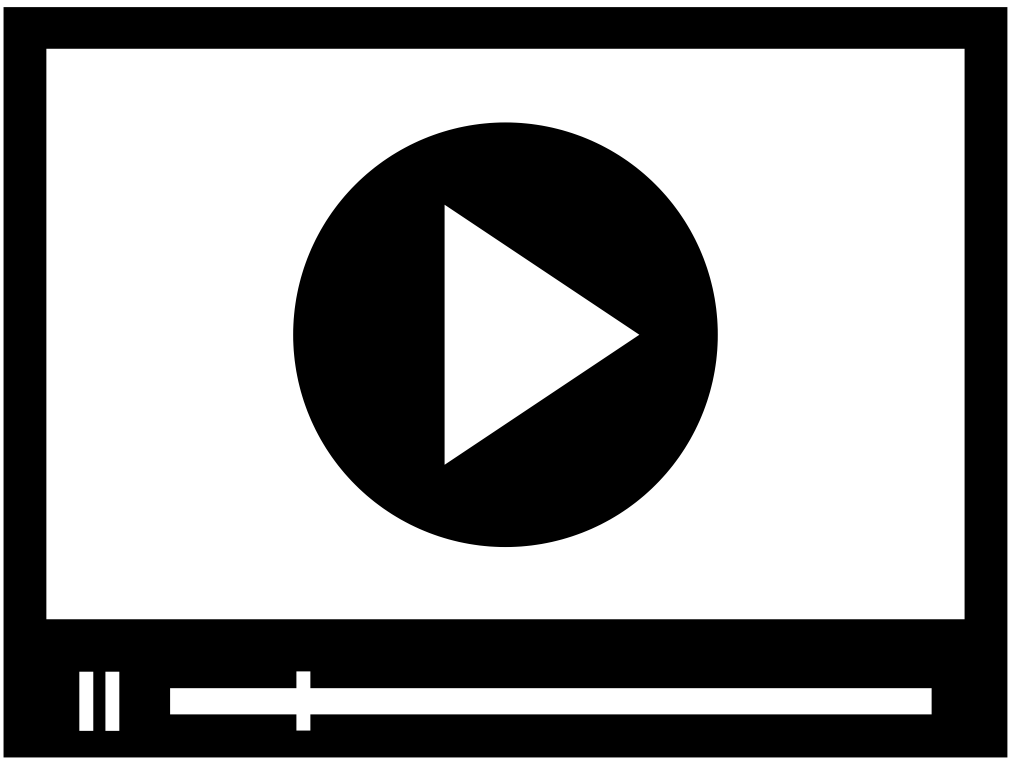
Significa que, além dos dados, o SBD contém a descrição completa de suas estruturas e restrições. Como se verá na arquitetura do SGBD, a descrição da estrutura e das restrições de dados, conhecida como metadados, isto é, dados que descrevem dados, é armazenada no catálogo do sistema de banco de dados.

ABSTRAÇÃO DE DADOS

Permite a representação conceitual dos dados por meio de modelos de dados que ocultam detalhes de armazenamento e implementação, os quais não interessam aos diferentes usuários, dando suporte a múltiplas visões lógicas dos dados e à independência de dados.

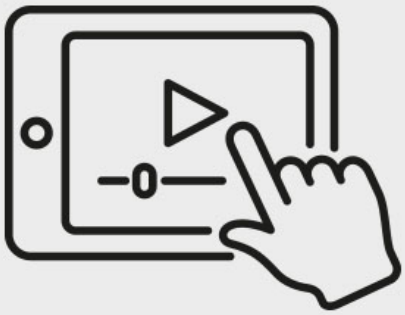
SUPORTE AO COMPARTILHAMENTO DE DADOS E PROCESSAMENTO DE TRANSAÇÕES CONCORRENTES

Permite que múltiplos usuários acessem o banco de dados simultaneamente.



Neste vídeo, o professor Sidney Ventury reforçará pontos importantes como os metadados e a arquitetura de três esquemas.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Os modelos de dados que suportam a abstração de dados e permitem sua independência lógica e física são classificados em modelos físicos, lógicos e conceituais.

MODELOS FÍSICOS

Descrevem como os dados são armazenados no computador mediante informações como tipos de arquivos, formatos e ordenação de registros, caminhos de acesso. São as várias formas de estruturas de arquivos que dependem do SGBD e do sistema operacional em que estão instaladas.

MODELOS LÓGICOS

São aqueles que representam, de maneira abstrata, a implementação dos bancos de dados, ocultando detalhes de como os dados são armazenados e acessados no disco. O modelo lógico mais tradicional, largamente utilizado na grande maioria dos sistemas de informação organizacionais, é o modelo relacional criado por Edgar Codd, mencionado no módulo anterior. A estrutura de dados do modelo relacional é a tabela (relação matemática) e deve ser estudada em profundidade em tema específico, assim como as implementações de SGBD relacional. Como vimos, há outros modelos lógicos de implementação do banco de dados entre aqueles que atualmente se denominam de modelos não relacionais ou NoSQL.

MODELOS CONCEITUAIS

São aqueles que representam a visão dos dados do ponto de vista do usuário final, no nível de abstração mais próximo do mundo real. Dentre esses, destaca-se o modelo de entidades e relacionamentos, criado pelo pesquisador Peter Chen, em 1976. O chamado modelo ER, segundo o próprio autor, foi criado para prover um melhor entendimento do modelo relacional, de modo a servir como uma etapa inicial no processo de projeto de banco de dados, denominada de modelagem conceitual dos dados.

Vale destacar que a modelagem conceitual de dados realizada com o modelo ER foi incorporada na linguagem UML (*Unified Modeling Language*), criada no final da década de

1990 para uniformizar a modelagem orientada a objetos sob a forma do modelo de classes, em que as entidades são representadas pelas classes de objetos e os relacionamentos pelas associações entre as classes. Na prática atual do processo de desenvolvimento de sistemas de informação, é comum a utilização do modelo de classes da UML como substituto do modelo ER nas fases iniciais do projeto de banco de dados.

📢 ATENÇÃO

Nesse ponto, vale a pena revisitar a definição de banco de dados citada no início do primeiro módulo, complementando-a com as principais características aqui mencionadas, Assim, podemos definir banco de dados como uma coleção autodescritiva de dados relacionados, com significado lógico inerente, que pode ser manipulada concorrentemente por usuários com visões diferentes.

VANTAGENS E DESVANTAGENS DA ABORDAGEM DE BANCO DE DADOS

A abordagem de SBD possui funcionalidades que conferem vantagens adicionais em relação a sistemas sem banco de dados, além das características que a diferenciam da abordagem de sistemas de arquivos. Algumas dessas funcionalidades do SBD são listadas a seguir e podem ser objeto de estudo em temas sobre implementação e administração de banco de dados.

CONTROLE DA REDUNDÂNCIA DE DADOS

Previne a possibilidade de inconsistência dos dados, a duplicação de esforço para manter os dados atualizados e o desperdício de espaço de armazenamento. O SGBD permite controlar o *trade off* entre o armazenamento em um único local no banco de dados *versus* a redundância forçada para melhorar o desempenho das consultas.

COMPARTILHAMENTO DE DADOS

Sendo os SBD multiusuários, realizam controle de concorrência de acesso aos dados compartilhados para garantir as propriedades de atomicidade, consistência, isolamento e durabilidade (propriedades ACID) das transações de banco de dados.

CONTROLE DE ACESSO

Mecanismos de segurança e autorização, como senhas para usuários e para grupos de usuários; restrição de acesso a partes do banco de dados; proibição de executar certas operações privilegiadas; acesso de usuário restrito apenas a transações autorizadas; proibição de uso de software privilegiado, como o software de administração do SBD.

MÚLTIPLAS INTERFACES DE USUÁRIOS E APLICAÇÕES

Provê diferentes linguagens de consulta para usuários casuais; linguagens de programação para programadores de aplicações; interfaces gráficas com formulários (telas) e menus para usuários de aplicações; interfaces para administração do banco de dados; interfaces de linguagem natural.

REPRESENTAÇÃO DE RELACIONAMENTOS ENTRE OS DADOS

Permite representar os relacionamentos existentes entre dados no mundo real, mediante mecanismos que dependem do modelo lógico de implementação do SBD.

CUMPRIMENTO DE RESTRIÇÕES DE INTEGRIDADE DOS DADOS

Previne a violação de restrições como tipo de dado ou domínio de valores admissíveis; unicidade de itens de dados por meio de chaves únicas; integridade referencial entre dados relacionados e restrições derivadas da semântica dos dados. Aqui também o SGBD permite controlar o *trade off* entre garantir o cumprimento automático das restrições ou deixar a sua especificação para os programas de aplicação.

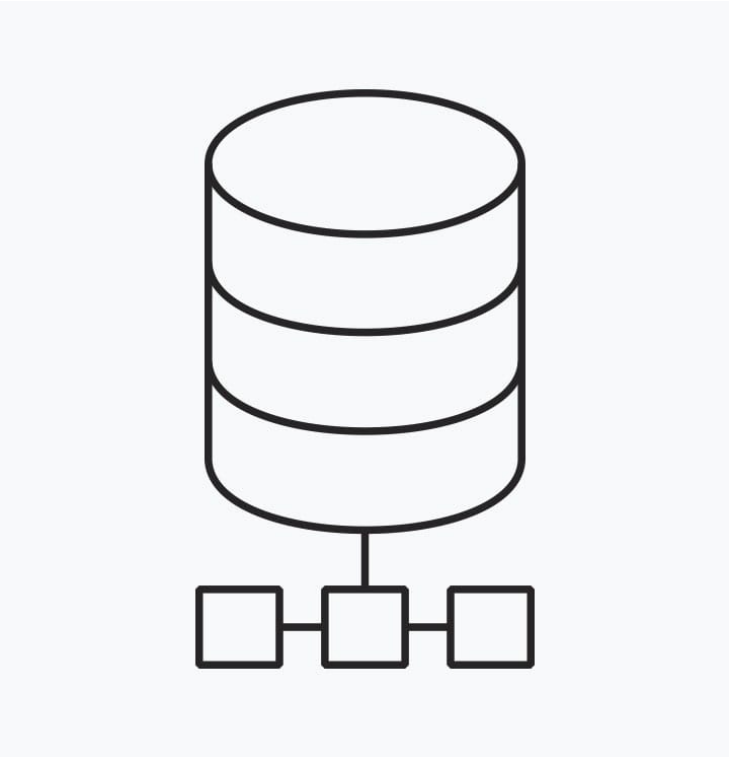
CAPACIDADE DE *BACKUP* E RECUPERAÇÃO DE DADOS:

Em caso de ocorrência de falhas de hardware ou de software, os mecanismos de cópia de segurança (*backup*) e posterior restauração (*recovery*) garantem a consistência de estado do banco de dados antes e depois da falha.

COMPARTILHAMENTO DE DADOS ENTRE MÚLTIPLOS USUÁRIOS SIMULTÂNEOS

É uma característica primordial do SBD e tem como resultado o controle de concorrência das transações.

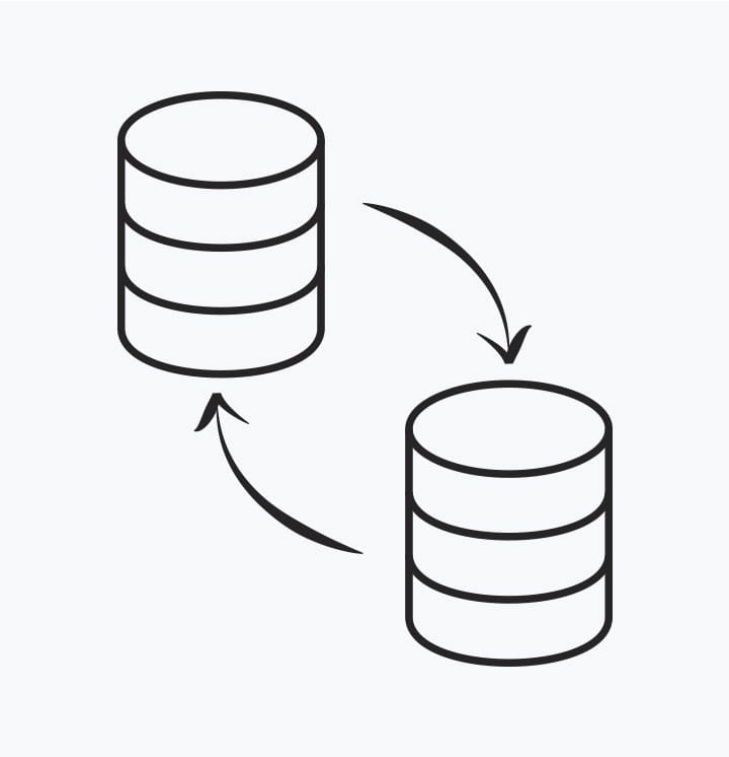
É responsabilidade do SGBD garantir as propriedades das transações, conhecidas pela sigla ACID, relaxando-as quando necessário para manter o desempenho sob seu controle e para não violar a integridade do banco de dados.



Fonte:Shutterstock

ATOMICIDADE (*ATOMICITY*)

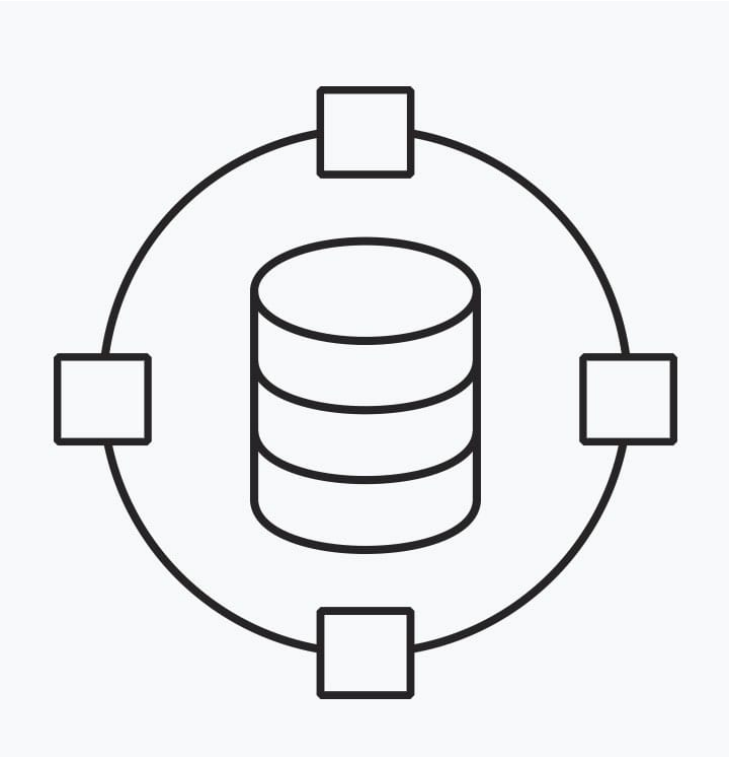
Cada transação é tratada como uma unidade composta de uma sequência de operações, de modo que deve executar completamente com sucesso ou falhar completamente.



Fonte:Shutterstock

CONSISTÊNCIA (*CONSISTENCY*)

Uma transação só pode levar o banco de dados de um estado válido para outro, de acordo com suas regras de integridade.



Fonte:Shutterstock

ISOLAMENTO (*ISOLATION*)

Cada transação é isolada das demais, isto é, essa propriedade assegura que transações executadas concorrentemente levem o banco de dados ao mesmo estado que chegaria se as transações fossem executadas sequencialmente.

Fonte:Shutterstock

DURABILIDADE (*DURABILITY*)

Uma vez que a transação é aceita (*committed*), o que significa que seu resultado foi gravado em memória não volátil, esse resultado permanecerá válido mesmo em caso de falhas do sistema.

autor/shutterstock

autor/shutterstock

VANTAGENS

As vantagens decorrentes dessas funcionalidades do SBD são várias: potencial para o estabelecimento de padrões de uso dos dados na organização; redução do tempo de desenvolvimento de aplicações; flexibilidade na manutenção dos dados; disponibilidade dos dados atualizados no âmbito de toda a organização; economia de escala, entre outras.

autor/shutterstock

DESVANTAGENS

Agora, vejamos as desvantagens da abordagem de banco de dados. Como já foi observado, a presença do SGBD como um software intermediário entre as aplicações e os dados armazenados provoca uma sobrecarga no desempenho do sistema como um todo. Além disso, há de se levar em conta o custo e o esforço adicional na capacitação e no oferecimento de funcionalidades sofisticadas como as já citadas anteriormente.

Assim, pode-se dizer que é preferível **não usar a abordagem de SBD** em algumas aplicações, tais como:

- Aplicações muito simples com dados estáticos e bem definidos, que se espera que não sejam alterados (exemplo: censo demográfico);

Aplicações de tempo real com requisitos rígidos os quais não possam ser atendidos com o *overhead* causado pelo SGBD (exemplo: controle de tráfego aéreo);

Sistemas embarcados, com poucos dados e com requisitos estritos de tempo real (exemplo: piloto automático);

Sistemas monousuários de uso sem concorrência, típicos de aplicações em *desktop* (exemplo: prontuário eletrônico de um único consultório médico).

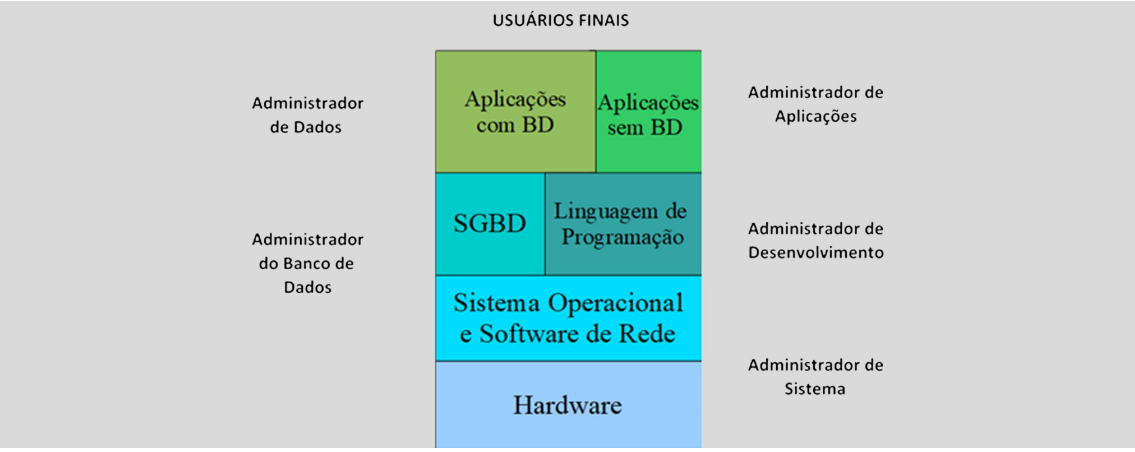
Fonte:Shutterstock

Fonte: Shutterstock | **Por**: Dmi T

Poderiam ser enquadradas nessa lista as aplicações que requerem dados volumosos com dinâmica não processável por SGBDs tradicionais, como a Internet das Coisas (IoT, de *Internet of Things*), porém, o advento de bancos de dados NoSQL, com modelos de dados específicos para tratamento de *Big Data*, busca a atender a esses tipos de aplicações incompatíveis com os modelos tradicionais de SGBD.

PAPÉIS EM SISTEMAS DE BANCOS DE DADOS

A figura a seguir mostra as camadas de um sistema de computação desde o hardware, onde são armazenados os arquivos com programas e dados, até as aplicações disponíveis aos usuários finais, que podem ser desenvolvidas com ou sem a utilização de um SGBD. Na ilustração, podemos ver também a existência de diferentes papéis de usuários ao longo das camadas de software.



Fonte: Tanaka (2018)

Em um sistema de computação corporativo de grandes organizações, é possível separar os diferentes papéis desempenhados como mostra a figura.

USUÁRIOS FINAIS

Beneficiários dos sistemas que rodam no sistema de computação, seja por aplicações desenvolvidas com a abordagem de banco de dados ou não.

ADMINISTRADOR DE APLICAÇÕES

Função técnico-gerencial responsável pela manutenção dos sistemas de aplicação e pelo suporte aos seus usuários, podendo ser exercida por vários administradores. Exemplo: administrador do sistema integrado de gestão empresarial, conhecido pela sigla ERP (*Enterprise Resource Planning*).

ADMINISTRADOR DE DESENVOLVIMENTO

Função técnica que pode ser desdobrada em equipes de desenvolvimento do sistema de aplicação, dependendo do porte e da complexidade do sistema, que se utilizam das ferramentas disponíveis no ambiente de desenvolvimento de sistemas.

ADMINISTRADOR DE DADOS

Função gerencial responsável pelo ambiente de dados da organização, que define políticas e responsabilidades sobre os recursos de dados, assim como as regras do negócio e os padrões

de dados a serem seguidos no desenvolvimento.

ADMINISTRADOR DO BANCO DE DADOS

Função técnica responsável pela criação e manutenção dos bancos de dados no SGBD, dando suporte às equipes de desenvolvimento no tocante aos objetos dos bancos de dados.

ADMINISTRADOR DE SISTEMA

Responsável por manter no ar o sistema de computação como um todo, com foco no hardware e no sistema operacional, bem como nas interfaces deste com os demais softwares instalados.

CABE OBSERVAR QUE NEM TODOS OS PAPÉIS DE ADMINISTRAÇÃO PODEM SER EXERCIDOS POR UMA ÚNICA PESSOA, MAS UMA PESSOA PODE EXERCER MAIS DE UM PAPEL. A DISTRIBUIÇÃO DESSES PAPÉIS POR PESSOAS E EQUIPES DEPENDERÁ DO PORTE DA ORGANIZAÇÃO E DO SISTEMA DE COMPUTAÇÃO, VARIANDO DESDE UMA ÚNICA PESSOA EXERCENDO TODOS OS PAPÉIS DE ADMINISTRAÇÃO ATÉ TODO UM DEPARTAMENTO DE TI ENCARREGADO DA ADMINISTRAÇÃO E DO SUPORTE AOS DIVERSOS SISTEMAS EM OPERAÇÃO.

VERIFICANDO O APRENDIZADO

1. ANALISE AS SEGUINTES AFIRMAÇÕES E RESPONDA QUAL ALTERNATIVA CORRESPONDE A CARACTERÍSTICAS QUE DISTINGUEM OS SISTEMAS DE BANCO DE DADOS DOS SISTEMAS DE ARQUIVOS.

I. NATUREZA AUTOCONTIDA DOS DADOS

II. VOLATILIDADE DOS DADOS
III. ABSTRAÇÃO DE DADOS

A) Somente as afirmações I e II.

B) Somente as afirmações I e III.

C) Somente as afirmações II e III.

D) Todas as afirmações estão corretas.

2. QUAL DAS ALTERNATIVAS ABAIXO NÃO É UMA FUNCIONALIDADE DOS SISTEMAS DE BANCO DE DADOS?

A) Representação de relacionamentos entre os dados.

B) Controle de redundância de dados.

C) Compartilhamento de dados.

D) Armazenamento de dados estáticos.

GABARITO

1. Analise as seguintes afirmações e responda qual alternativa corresponde a características que distinguem os sistemas de banco de dados dos sistemas de arquivos.
I. Natureza autocontida dos dados
II. Volatilidade dos dados
III. Abstração de dados

A alternativa **"B "** está correta.

A natureza autocontida dos dados, com o armazenamento junto aos metadados, e a abstração de dados, permitindo a independência dos dados, são características que diferenciam os SBD dos sistemas de arquivos. Já a volatilidade dos dados não é uma característica dos SBD.

2. Qual das alternativas abaixo não é uma funcionalidade dos sistemas de banco de dados?

A alternativa **"D "** está correta.

O armazenamento de dados estáticos é uma das situações em que não é recomendável o uso da abordagem de banco de dados, pois não se trata de uma funcionalidade típica dos SBDs, como as demais alternativas.

MÓDULO 3

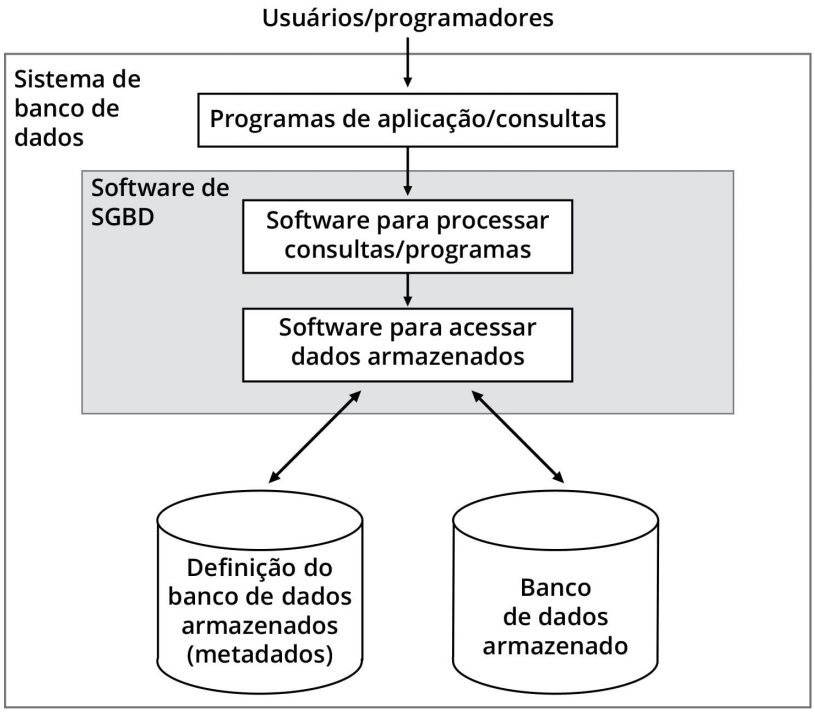
- Descrever a arquitetura dos sistemas de gerência de banco de dados (SGBD)

COMPONENTES DE UM SISTEMA DE BANCO DE DADOS

A figura a seguir mostra uma visão simples do ambiente de banco de dados, compreendendo programas de aplicação ou consultas de usuários que acessam os dados e metadados armazenados em disco através do SGBD. Este, por sua vez, compõe-se, simplificadamente, de um software para processar consultas e programas e outro para acessar os dados e metadados armazenados.

Nota-se, na figura, que não se confundem o SBD e o SGBD, visto que este é um componente daquele que engloba também os programas e as consultas, bem como os dados e metadados armazenados.

A fronteira do SBD engloba os programas que implementam as aplicações, bem como as consultas provenientes de usuários com acesso a linguagens e interfaces de consulta. O SGBD é o software intermediário, uma caixa preta a ser aberta no próximo módulo. A fronteira do SBD envolve os metadados armazenados no catálogo (às vezes chamado de dicionário de dados) e o próprio conteúdo armazenado no banco de dados.



Fonte: Elmasri e Navathe (2019)

COSTUMA-SE DEFINIR METADADOS COMO O ESQUEMA DO BANCO DE DADOS, ESTRUTURADO DE ACORDO COM O MODELO LÓGICO DE IMPLEMENTAÇÃO. ASSIM, MODELAR UM BANCO DE DADOS CONSOANTE AO MODELO LÓGICO DE IMPLEMENTAÇÃO EQUIVALE A ESQUEMATIZAR O BANCO DE DADOS CONFORME OS CONSTRUTORES DESSE MODELO. POR EXEMPLO, NO MODELO RELACIONAL, O ESQUEMA É COMPOSTO POR TABELAS E SUAS COLUNAS. CADA COMANDO DE DEFINIÇÃO DE DADOS, CRIANDO, ALTERANDO OU REMOVENDO UMA TABELA, PROVOCA UMA MUDANÇA NO ESQUEMA DO BANCO DE DADOS.

Por outro lado, chama-se de estado ou instância o conteúdo do banco de dados armazenado em um momento. Cada manipulação no banco de dados mediante comandos de inserção, atualização ou remoção de dados provoca uma mudança de estado, gerando uma nova instância do banco de dados.

CURIOSIDADES SOBRE BANCO DE DADOS

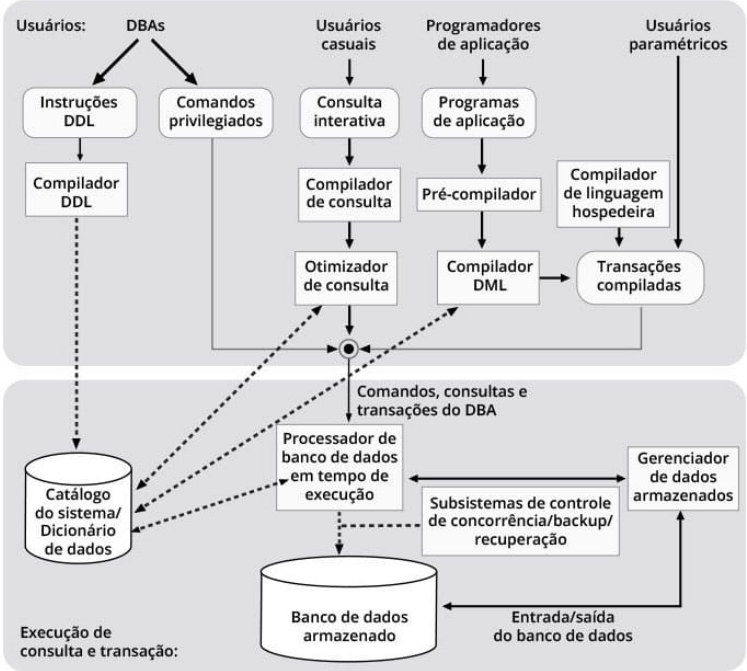
Como vimos, o termo **banco de dados** é correntemente usado com o sentido do original, em inglês, *database*, cuja origem, segundo o *Oxford English Dictionary*, remonta a 1962 num relatório de uma empresa na Califórnia. Raras são as referências ao termo *data bank* como, por exemplo, no mencionado artigo de Edgar Codd sobre o modelo relacional (CODD, 1970).

Em espanhol, usa-se *base de datos*, no francês *base de données*, enquanto em alemão se diz *Datenbank*, e em italiano *banca dati*.

Em português, existem os dois termos, sendo banco de dados usado no sentido geral do ambiente que engloba o sistema, como na figura anterior, enquanto base de dados tem o sentido mais restrito do conteúdo do banco, isto é, corresponde ao estado ou à instância do banco de dados. Por exemplo, costuma-se referir à base de dados da Receita Federal como o conjunto de dados armazenados sobre os contribuintes e não como o sistema que gerencia os dados.

MÓDULOS DE UM SISTEMA DE GERÊNCIA DE BANCO DE DADOS

A figura a seguir ilustra, em detalhes, os diversos módulos que compõem um SGBD, sendo a parte superior correspondente ao processamento das consultas e aplicações, e a parte inferior ao acesso a metadados e dados armazenados (a base de dados).



Fonte: Elmasri e Navathe (2019)

Vamos conferir a descrição desses módulos.

USUÁRIOS

Da esquerda para a direita, vemos os usuários, começando com os administradores de banco de dados (ABD, do inglês DBA – *Database Administrator*).

ABD

O ABD usa comandos da linguagem de definição de dados (LDD, do inglês DDL – *Data Definition Language*) para criar, alterar ou remover objetos do banco de dados (comandos CREATE, ALTER, DROP no padrão SQL), os quais ficam armazenados no catálogo do sistema, que contém os metadados.

O ABD também possui privilégio para executar comandos de controle de dados, conhecidos por alguns autores como linguagem de controle de dados (LCD, do inglês DCL – *Data Control Language*), para conceder e revogar permissões de acesso aos dados (comandos GRANT e REVOKE no padrão SQL), entre outros. Note que o destino desses comandos privilegiados é um nó central do SGBD, por onde passam todos os comandos destinados ao processador de banco de dados em tempo de execução, denominado de processador de *runtime* ou *runtime engine*.

USUÁRIOS CASUAIS

Seguindo para a direita, vemos os usuários casuais, os quais fazem consultas interativas através de uma interface para consultas *ad hoc*, ou seja, consultas não programadas

previamente. As consultas são feitas tipicamente com o comando SELECT no padrão SQL do modelo relacional e provido de modo similar em outros modelos de implementação de banco de dados. Esses comandos são compilados por um compilador da linguagem de consulta e passam por um otimizador de consulta antes de chegar ao nó central do SGBD.

PROGRAMADORES DE APLICAÇÕES

Mais à direita, aparecem os programadores de aplicações que escrevem os programas em uma linguagem de programação hospedeira, como por exemplo, Java, PHP ou Python, nos quais estão embutidos comandos de consulta (SELECT), inserção, atualização e exclusão de dados em uma linguagem de manipulação de dados (LMD, do inglês DML – *Data Manipulation Language*).

No padrão SQL, esses comandos são, respectivamente, INSERT, UPDATE e DELETE. Note que eles manipulam dados, diferentemente dos comandos da linguagem de definição de dados (CREATE, ALTER, DROP), que manipulam metadados.

PROGRAMAS DE APLICAÇÃO

Os programas de aplicação, portanto, possuem comandos híbridos, da linguagem hospedeira e da linguagem de consulta e manipulação de dados.

PRÉ-COMPILADOR

Os programas de aplicação são processados, inicialmente, por um pré-compilador, responsável por separar os comandos e os repassar para os compiladores das respectivas linguagens.

COMPILADORES

Cabe a esses compiladores produzir o código das aplicações sob a forma de transações executáveis, que ficam à disposição dos usuários paramétricos.

USUÁRIOS PARAMÉTRICOS

Os usuários paramétricos são assim chamados porque interagem com o sistema através de parâmetros passados em interfaces apropriadas. Por exemplo, um agente de viagens faz uma reserva de passagem aérea passando ao sistema os dados do passageiro, data e hora da viagem, número do voo, número do assento e outros parâmetros necessários para efetivar a reserva.

TRANSAÇÕES COMPILADAS

Uma vez executadas as **transações compiladas**, assim como os demais comandos provenientes de usuários ou aplicações, são passadas ao nó central do SGBD para posterior

processamento do processador de *runtime*.

Atenção! Antes de prosseguir na parte de baixo da figura, note que alguns processos da parte de cima, como o otimizador de consulta e o compilador da LMD, estão ligados ao catálogo por linhas tracejadas, que denotam fluxos de controle, enquanto as linhas cheias representam fluxos de dados, além de controle. Isso é necessário porque as referências a objetos do banco de dados existentes no catálogo devem ser consistentes com os objetos, criados e mantidos pelo ABD mediante comandos da LDD.

PROCESSAMENTO DO ACESSO AOS DADOS ARMAZENADOS

Prosseguindo para a parte do processamento do acesso aos dados armazenados, observamos que o processador de *runtime* é o coração do SGBD. De fato, o diferencial dos produtos de SGBD reside na eficiência e na funcionalidade desse processador, segredo industrial em muitos dos SGBDs proprietários, a ponto de exigir rigorosos termos de confidencialidade das pessoas que têm acesso ao seu código.

ACESSO A BASE DE DADOS

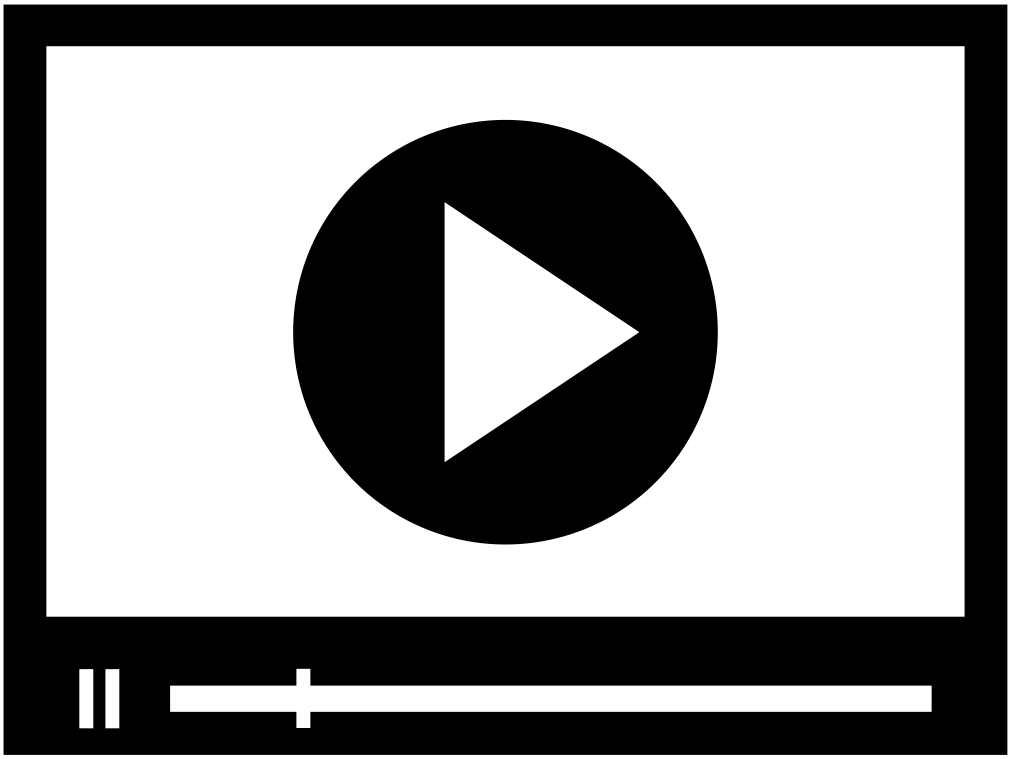
O processador de *runtime* também precisa acessar e, por vezes, modificar o catálogo, dependendo da natureza dos comandos, das consultas ou transações que estiver processando. Como mostra a figura, ele acessa a base de dados diretamente, sob controle de subsistemas de controle de concorrência, *backup* e *recovery*. Quando precisa realizar operações de entrada e saída (gravação e leitura) na base de dados, o processador de *runtime* se vale de um gerenciador de dados armazenados.

A descrição dada, embora simplificada, denota a complexidade de um SGBD, software que evoluiu desde a década de 1960 a ponto de se tornar um recurso robusto, eficiente e indispensável para a maioria das organizações privadas ou públicas, principalmente com base no modelo relacional de banco de dados.

Existe uma grande quantidade de SGBDs disponíveis no mercado, como mostrou o DB-Engines Ranking, ao listar mais de 300 SGBDs, sendo cerca de 130 deles do modelo relacional.

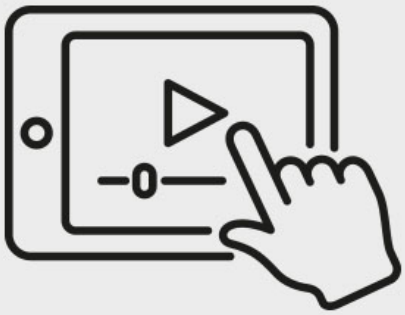
SAIBA MAIS

A excelente entrada na Wikipédia, intitulada *Comparison of relational database management systems*, faz uma comparação de diversas características de SGBDs relacionais, contendo informações detalhadas sobre mais de 60 produtos.



Neste vídeo, o professor Sidney Ventury abordará pontos importantes como *runtime*, transações compiladas, entre outros.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



UM EXEMPLO DE SGBD RELACIONAL: POSTGRESQL

Como vimos anteriormente na evolução histórica dos SGBDs, o PostgreSQL originou-se do projeto Ingres da Universidade da Califórnia em Berkeley, na década de 1970, tendo sido

sucessivamente renomeado de Postgres (*Post Ingres*), depois Postgres 95 e, finalmente, PostgreSQL.

Embora fossem implementações do modelo relacional proposto por Edgard Codd (CODD, 1970), o Ingres e o Postgres originalmente usavam uma linguagem diferente da SQL, adotada pela IBM, conhecida como QueL (*Query Language*). Por muitos anos, a QueL persistiu nesses SGBDs em concorrência direta com a SQL da IBM, até que o ANSI (*American National Standard Institute*) e depois a ISO (*International Standard Organization*) reconheceram a SQL como a linguagem padrão para o banco de dados relacional. A partir de então, a SQL foi implementada no Postgres, passando a ser chamada de PostgreSQL.

O PostgreSQL, além de ser reconhecido como *the world's most advanced open source relational database*, possui uma completa documentação igualmente reconhecida como referência global para os conceitos de SGBD relacional. Não é sem razão que o PostgreSQL, por ser *open source* e ter licença livre, é vastamente utilizado no ensino de banco de dados, além de ser adotado comercialmente em grandes organizações ao redor do mundo.

O POSTGRESQL É UM TÍPICO SGBD RELACIONAL-OBJETO, ISTO É, UM SGBD CUJA ESTRUTURA DE DADOS BÁSICA É A RELAÇÃO (TABELA), PORÉM CONTEMPLA EXTENSÕES DE TIPOS DE DADOS E CARACTERÍSTICAS PRÓPRIAS DA ORIENTAÇÃO A OBJETOS. ASSIM, POSSUI SUPORTE EM GRANDE PARTE AO PADRÃO SQL E IMPLEMENTA EXTENSÕES ÚTEIS COMO: CONSULTAS COMPLEXAS, GATILHOS (*TRIGGERS*), VISÕES MATERIALIZADAS ATUALIZÁVEIS, CONTROLE DE CONCORRÊNCIA MULTIVERSIONADO. ALÉM DISSO, PODE SER ESTENDIDO DE MUITAS FORMAS, POR EXEMPLO, GERANDO NOVOS TIPOS DE DADOS, FUNÇÕES GENÉRICAS E AGREGADAS, OPERADORES,

MÉTODOS DE INDEXAÇÃO E LINGUAGENS PROCEDURAIS.

Em jargão de sistemas, o PostgreSQL usa um modelo de computação cliente/servidor, assim como a maioria dos SGBDs relacionais empresariais. Uma sessão PostgreSQL consiste dos seguintes processos cooperativos (programas):

Back end

Um processo servidor (*back end*), responsável por gerenciar os arquivos do banco de dados, aceitar conexões por aplicações clientes e executar ações sobre o SBD em nome dos clientes. O programa servidor é chamado postgres.

Front end

As aplicações clientes de usuários (*front end*) que desejam executar operações no banco de dados. Aplicações clientes podem ser diversas em natureza, como uma ferramenta orientada a texto, uma aplicação gráfica, um servidor Web que acessa o banco de dados para exibir páginas Web, ou uma ferramenta especializada de manutenção. Algumas aplicações clientes são fornecidas com a distribuição PostgreSQL (psql, pgadmin).

Fonte:Shutterstock

Fonte: Shutterstock | **Por**: Ton Snoei

O PostgreSQL roda em todos os sistemas operacionais importantes, incluindo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) e Windows. Possui todas as propriedades ACID das transações (atomicidade, consistência, isolamento e durabilidade) e tem completo suporte para chaves estrangeiras, junções, visões, funções, *triggers*, e procedimentos armazenados em múltiplas linguagens de programação, incluindo Java, Perl, Python, Ruby, Tcl, C/C++, e sua própria PL/pgSQL, similar à PL/SQL do Oracle.

Como banco de dados de classe empresarial, PostgreSQL possui características sofisticadas como controle de concorrência multiversão (*Multi Version Concurrency Control*), recuperação "*point in time*" (PiTR – *Point in Time Recovery*), *tablespaces*, replicação assíncrona, transações aninhadas (*savepoints*), *online/hot backups*, um sofisticado planejador/otimizador de consultas, e *write ahead logging* (WAL) para tolerância a falhas.

O PostgreSQL é altamente escalável tanto na quantidade de dados que pode gerenciar como no número de usuários concorrentes que pode acomodar. Existem sistemas PostgreSQL em ambientes de produção que gerenciam centenas de usuários simultâneos com dezenas de *terabytes* de dados armazenados.

Além de possuir um catálogo de sistema totalmente relacional, capaz de suportar múltiplos *schemas* por *database*, o catálogo do PostgreSQL é também acessível através do *Information Schema*, um conjunto de visões de metadados definido no padrão SQL.

Fonte:Shutterstock

Fonte: Shutterstock | **Por**: Alexander Supertramp

ATENÇÃO

PostgreSQL suporta índices compostos, parciais e funcionais que podem usar seus métodos de armazenamento B-tree, R-tree, hash ou GiST. Este último é um sofisticado *framework* que serve de base para muitos projetos públicos que usam PostgreSQL, como o OpenFTS (Open Source Full Text Search engine), que provê indexação online de dados e ranking de relevância para pesquisa em bancos de dados, e o PostGIS, um projeto que adiciona suporte para objetos geográficos, permitindo seu uso como banco de dados espacial para Sistemas Geográficos de Informação.

Outras características avançadas do PostgreSQL incluem herança de tabelas, sistema de regras e eventos de banco de dados.

HERANÇA DE TABELAS

Coloca um "sabor" orientado a objeto na criação de tabelas, permitindo a projetistas de banco de dados derivar novas tabelas de outras tabelas, tratando-as como classe base. Esse esquema suporta tanto herança simples como múltipla.

SISTEMA DE REGRAS

Também chamado de sistema de reescrita de consultas, permite ao projetista de banco de dados criar regras que identificam operações específicas para uma dada tabela ou visão e dinamicamente transformá-las em operações alternativas quando são processadas (cláusula INSTEAD OF).

SISTEMA DE EVENTOS

É um sistema de comunicação interprocessos em que mensagens e eventos podem ser transmitidos entre clientes usando os comandos LISTEN e NOTIFY, permitindo tanto comunicação simples *peer to peer* como avançadas coordenações entre eventos de banco de dados. Clientes PostgreSQL podem monitorar eventos de banco de dados, como atualizações, inserções e exclusões em tabelas, enquanto eles acontecem.

📢 ATENÇÃO

O código fonte do PostgreSQL é disponível sob a licença *open source* BSD. Essa licença dá a liberdade de usar, modificar e distribuir o PostgreSQL da forma que se desejar, com código aberto ou fechado. Quaisquer modificações, melhoramentos ou mudanças que você faça serão seus para fazer o que quiser. Por isso, o PostgreSQL não é somente um poderoso SGBD capaz de rodar o negócio da empresa. Também é uma plataforma de desenvolvimento sobre a qual se pode desenvolver produtos de software *in-house* que requeiram um suporte adequado a banco de dados.

Com a grande aceitação do PostgreSQL no mercado, desde 2004, um grupo de contribuidores do seu desenvolvimento fundou a empresa Enterprise DB, que fornece uma versão empresarial de mesmo nome, com consideráveis extensões ao PostgreSQL.

VERIFICANDO O APRENDIZADO

1. QUAL É O PRINCIPAL MÓDULO COMPONENTE DE UM SGBD, PELO QUAL PASSAM TODAS AS CONSULTAS E TRANSAÇÕES DE USUÁRIOS, SEJAM ADMINISTRADORES, USUÁRIOS CASUAIS OU PARAMÉTRICOS?

A) Compilador de linguagem de manipulação de dados.

B) Catálogo de metadados.

C) Otimizador de consulta.

D) Processador de *run time*.

2. QUAL É O MODELO DE COMPUTAÇÃO UTILIZADO PELO POSTGRESQL, ASSIM COMO PELA MAIORIA DOS SGBDS RELACIONAIS EMPRESARIAIS?

A) Centralizado.

B) Pessoal.

C) Cliente/Servidor.

D) *Stand Alone*.

GABARITO

1. Qual é o principal módulo componente de um SGBD, pelo qual passam todas as consultas e transações de usuários, sejam administradores, usuários casuais ou paramétricos?

A alternativa **"D "** está correta.

O processador do banco de dados em tempo de execução (*run time*) recebe todas as requisições dos usuários e as processa para acessar o catálogo e a base de dados.

2. Qual é o modelo de computação utilizado pelo PostgreSQL, assim como pela maioria dos SGBDs relacionais empresariais?

A alternativa **"C "** está correta.

O PostgreSQL usa um modelo de computação cliente/servidor, em que uma sessão consiste de um processo servidor (*back end*), que gerencia os arquivos do banco de dados, aceita conexões por aplicações clientes e executa ações sobre o banco de dados em nome dos

clientes, e aplicações clientes de usuários (*front end*) que desejam executar operações no banco de dados.

CONCLUSÃO

CONSIDERAÇÕES FINAIS

Este tema abordou uma introdução aos sistemas de banco de dados, resumindo sua evolução histórica desde a invenção do disco magnético, que possibilitou o início da era de processamento de dados por computador.

Foram examinadas as principais características do sistema de banco de dados (SBD), com ênfase naquelas que o diferenciam dos sistemas de arquivos, bem como nas suas funcionalidades, vantagens e desvantagens.

Finalizamos o tema com a descrição da arquitetura de um sistema de gerência de banco de dados (SGBD), componente central do SBD, terminando com um exemplo de SGBD relacional, amplamente utilizado no ensino e no mercado, o PostgreSQL.

Encerraremos este tema aprofundando um pouco mais sobre sistema de banco de dados.

Para ouvir um *podcast* sobre o assunto, acesse a versão online deste conteúdo.

REFERÊNCIAS

BALIEIRO, R. **Banco de Dados**. 1. ed. Rio de Janeiro: SESES, 2015

BOOCH, G; RUMBAUGH, J; JACOBSON, I. **The Unified Modeling Language User Guide**. Reading: Addison-Wesley. 1998.

CHEN, P. **The Entity–Relationship Model – Toward A Unified View of Data**. *In*: ACM Transactions on Database Systems. 1 (1): 9–36, 1976.

CODD, E. F. **A Relational Model of Data for Large Shared Data Banks**. *In*: Communications of the ACM. 13 (6): 377–387, 1970.

DB-ENGINES. **DB-Engines Ranking of Relational DBMS**. Consultado em meio eletrônico em: 3 jun. 2020.

ELMASRI, R.; NAVATHE, S. **Sistemas de Banco de Dados**. 7. ed. São Paulo: Pearson, 2019.

IBM. **SQL: The language of Db2**.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 9075-1:2016**: Information technology - Database languages - SQL - Part 1: Framework (SQL/Framework). Publicado em: dez. 2016.

KLUG, A.; TSICHRITZIS, D. The ANSI/X3/SPARC DBMS framework: report of the study group on database management systems. *In*: **Information Systems. Data: creations, management and utilization**. v. 3, 3. Elsevier Journal, 1978.

MELO, R; SILVA, S.; TANAKA, A. **Banco de Dados em Aplicações Cliente-Servidor**. Rio de Janeiro: Infobook, 1998.

PPLWARE. **Charles Babbage – O pioneiro dos computadores**. Publicado em: 28 set. 2017.

POSTGRESQL. PostgreSQL: o banco de dados relacional de código aberto mais avançado do mundo. Consultado em meio eletrônico em: 3 jun. 2020.

TANAKA, A. **Notas de Aula sobre Banco de Dados do professor Asterio Tanaka**. Disponível sob licença Creative Commons BR Atribuição – CC BY, 2018.

WIKIPEDIA. **Comparison of relational database management systems**. Consultado em meio eletrônico em: 6 jun. 2020.

YUGE, C. **Crise da COVID-19 aumenta a procura por programadores de COBOL**. *In*: Canaltech. Publicado em: 13 abr. 2020.

EXPLORE+

Para reforçar o conteúdo visto, todo bom livro-texto de fundamentos de banco de dados possui uma introdução ao assunto, nos moldes deste tema. Recomendamos o livro-texto mais adotado mundialmente: *Sistemas de Banco de Dados*, de Ramez Elmasri e Shamkant B. Navathe. Vale a leitura da parte 1, sobre introdução a banco de dados, formada pelos capítulos: 1 – Bancos de Dados e Usuários de Bancos de Dados; 2 – Conceitos e Arquitetura do Sistema de Banco de Dados

Sugerimos uma visita ao excelente *DB-Engines Ranking*, que possui uma vasta classificação dos SGBDs mais populares, nos diversos modelos de dados, com breves explicações sobre cada modelo e informações sobre cada produto.

Como dissemos, a documentação do PostgreSQL, disponível em website de mesmo nome, é reconhecidamente um material de referência para o modelo relacional-objeto em geral, servindo como excelente ponto de partida para o entendimento do tema. Sugerimos iniciar pela seção *About* e prosseguir fazendo o *Tutorial*, que inclui *Getting Started*, *The SQL Language* e *Advanced Features*.

Leia a matéria *Crise da COVID-19 aumenta a procura por programadores de COBOL*, de Claudio Yuge.

Como dissemos, o PostgreSQL é reconhecido como como o mais avançado SGBD *open source* do mundo. Acesse o website do PostgreSQL, onde você encontrará a possibilidade de baixá-lo e saber sobre as últimas atualizações.

Leia o artigo *The ANSI/X3/SPARC DBMS framework report of the study group on database management systems*, de Dennis Tsichritzis e Anthony Klug, e aprofunde seus conhecimentos sobre a arquitetura ANSI/SPARC.

Acesso o site da Wikipedia e busque por *Comparison of relational database management systems*, que traz uma excelente comparação de diversas características de SGBDs

relacionais, contendo informações detalhadas sobre mais de 60 produtos.

Acesse o website da ISO (*International Standard Organization*) e busque pela resolução ISO/IEC 9075-1:2016, na qual a SQL é reconhecida como a linguagem padrão para o banco de dados relacional.

CONTEUDISTA

Nathielly de Souza Campos

🔗 CURRÍCULO LATTES